Apple IIc
#1: Mouse Differences on IIe and IIc

Revised by: Matt Deatherage November 1988
Revised by: Cameron Birse February 1986

This Technical Note explains differences between the IIe and IIc when working with a mouse and how to write programs which function properly on both machines.

If you use the mouse firmware routines (i.e., `SetMouse`) to control the mouse, then these routines will perform the same function on the IIc as they do on the IIe. However, a program which uses the mouse may not behave the same on both computers, and there are two reasons for the possible differences.

If a program does not properly set the environment prior to calling the mouse firmware routines, it is possible for a program to work on one machine and not the other. In addition, there are differences in machines and although the ROM routines perform the same functions, there may be a noticeable difference in the mouse behavior between the two machines.

This Note explains the fundamental differences between the way the mouse works on the two machines. We point out precautions that you need to take to ensure your assembly language programs work properly on both machines. (With the exception of mouse movement scaling described below, neither BASIC nor Pascal programs need be concerned with setting the proper environment.)

The Apple IIe mouse card has a microprocessor on it which constantly polls the mouse to get status and position information. This data is kept on the card and is available whenever the program requests it through the `ReadMouse` routine. If the mouse is in passive mode, this information will be picked up by the main program whenever it gets around to it.

The `SetMouse` routine can set the mouse card to issue interrupts under certain conditions. When the mouse card determines that such conditions exist, it issues an interrupt. This interrupt stops the main computer and goes to whatever interrupt handling routine has been prepared. This routine then reads the information from where the card processor saved it and puts it in the screen holes. When using a mouse on an Apple with a mouse card, your program is only interrupted if you have requested it, and the data in the screen holes is changed only when the program's interrupt handler or polling routine calls `ReadMouse`. In addition, enabling and disabling interrupts does not affect the card's microprocessor from updating the mouse information.

The Apple IIc mouse does not have a card microprocessor, so mouse information is collected by interrupting the microprocessor of the IIc itself. When the interrupt occurs, the firmware captures it and processes it, which includes updating the screen holes. The interrupt is passed only if `SetMouse` set up the conditions to do so.

Having the mouse interrupt the computer's microprocessor also means that your program is being constantly interrupted, which affects program timing. This interruption also means that the screen holes are constantly updated with X and Y information, even in passive mode, since this information must be stored somewhere and there is no card to keep it in. If you have disabled interrupts, the mouse can never interrupt the microprocessor, so the X and Y values are never updated and calling `ReadMouse` will indicate that there has been no mouse movement.

Since the Apple IIc is constantly interrupted while the mouse is on, the program's performance may be affected. To minimize this effect, the IIc responds one-half as frequently to mouse movements as does the mouse card, which means the mouse must be moved twice as far to create the same on-screen effect. If you want the same behavior on both machines, multiply the IIc X and Y values by two and the clamping value by one half. You do not need to make any changes to these values if your program is running on a IIe.

With this exception for mouse movement, your assembly language program can ignore which machine it is running on by following the precautions listed in Mouse Technical Note #1, Interrupt Environment with the Mouse (you must take these conditions into account if you want your assembly language program to behave similarly on both machines). If you are working in BASIC or Pascal, these conditions are already handled for you.

Further Reference

- *Apple IIc Technical Reference Manual, Second Edition*
- Mouse Technical Note #1, Interrupt Environment with the Mouse